Express Mail Label No. EL601699713US 07-19-00 A

UTILITY PATENT APPLICATION TRANSMITTAL
(Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.
POU9-2000-0091-US1
Total Pages in this Submission

jc867 U.S. PTO 07/18/00

jc814 U.S. PTO 09/618420 07/18/00

TO THE ASSISTANT COMMISSIONER FOR PATENTS
Box Patent Application
Washington, D.C. 20231

Transmitted herewith for filing under 35 U.S.C. 111(a) and 37 C.F.R. 1.53(b) is a new utility patent application for an invention entitled:

DETECTING WHEN TO PREFETCH INODES AND THEN PREFETCHING INODES IN PARALLEL

and invented by:

Frank B. Schmuck, James C. Wyllie

If a CONTINUATION APPLICATION, *check appropriate box and supply the requisite information:*

☐ **Continuation** ☐ **Divisional** ☐ **Continuation-in-part (CIP)** of prior application No.: _____

Which is a:

☐ **Continuation** ☐ **Divisional** ☐ **Continuation-in-part (CIP)** of prior application No.: _____

Which is a:

☐ **Continuation** ☐ **Divisional** ☐ **Continuation-in-part (CIP)** of prior application No.: _____

Enclosed are:

Application Elements

1. ☒ Filing fee as calculated and transmitted as described below
2. ☒ Specification having **38** pages and including the following:
   a. ☒ Descriptive Title of the Invention
   b. ☐ Cross References to Related Applications *(if applicable)*
   c. ☐ Statement Regarding Federally-sponsored Research/Development *(if applicable)*
   d. ☐ Reference to Microfiche Appendix *(if applicable)*
   e. ☒ Background of the Invention
   f. ☒ Brief Summary of the Invention
   g. ☒ Brief Description of the Drawings *(if drawings filed)*
   h. ☒ Detailed Description
   i. ☒ Claim(s) as Classified Below
   j. ☒ Abstract of the Disclosure

UTILITY PATENT APPLICATION TRANSMITTAL
(Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No. POU9-2000-0091-US1
Total Pages in this Submission

Application Elements (Continued)

3. ☒ Drawing(s) *(when necessary as prescribed by 35 USC 113)*
   a. ☒ Formal Number of Sheets **Five (5)**
   b. ☐ Informal Number of Sheets _____
4. ☒ Oath or Declaration
   a. ☐ Newly executed *(original or copy)* ☒ Unexecuted
   b. ☐ Copy from a prior application (37 CFR 1.63(d)) *(for continuation/divisional application only)*
   c. ☒ With Power of Attorney ☐ Without Power of Attorney
   d. ☐ *DELETION OF INVENTOR(S)*
   Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. 1.63(d)(2) and 1.33(b).
5. ☐ Incorporation By Reference *(usable if Box 4b is checked)*
   The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. ☐ Computer Program in Microfiche *(Appendix)*
7. ☐ Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all must be included)*
   a. ☐ Paper Copy
   b. ☐ Computer Readable Copy *(identical to computer copy)*
   c. ☐ Statement Verifying Identical Paper and Computer Readable Copy

Accompanying Application Parts

8. ☐ Assignment Papers *(cover sheet & document(s))*
9. ☐ 37 CFR 3.73(B) Statement *(when there is an assignee)*
10. ☐ English Translation Document *(if applicable)*
11. ☒ Information Disclosure Statement/PTO-1449 ☒ Copies of IDS Citations
12. ☐ Preliminary Amendment
13. ☒ Acknowledgment postcard
14. ☒ Certificate of Mailing
    ☐ First Class ☒ Express Mail *(Specify Label No.):* **EL601699713US**

UTILITY PATENT APPLICATION TRANSMITTAL
(Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.
POU9-2000-0091-US1

Total Pages in this Submission

Accompanying Application Parts (Continued)

15. ☐ Certified Copy of Priority Document(s) *(if foreign priority is claimed)*

16. ☐ Additional Enclosures *(please identify below):*

Fee Calculation and Transmittal

CLAIMS AS FILED

For	#Filed	#Allowed	#Extra	Rate	Fee
Total Claims	69	- 20 =	49	x $18.00	$882.00
Indep. Claims	12	- 3 =	9	x $78.00	$702.00
Multiple Dependent Claims (check if applicable) ☐					$0.00
				BASIC FEE	$690.00
OTHER FEE *(specify purpose)*					$0.00
				TOTAL FILING FEE	$2,274.00

☐ A check in the amount of to cover the filing fee is enclosed.

☒ The Commissioner is hereby authorized to charge and credit Deposit Account No. **09-0463 (IBM)** as described below. A duplicate copy of this sheet is enclosed.

- ☒ Charge the amount of **$2,274.00** as filing fee.
- ☒ Credit any overpayment.
- ☒ Charge any additional filing fees required under 37 C.F.R. 1.16 and 1.17.
- ☐ Charge the issue fee set in 37 C.F.R. 1.18 at the mailing of the Notice of Allowance, pursuant to 37 C.F.R. 1.311(b).

Blanche E. Schiller
Signature

Dated: **July 18, 2000**

Blanche E. Schiller, Esq.
Reg. No. 35,670
HESLIN & ROTHENBERG, P.C.
5 Columbia Circle
Albany, NY 12203
Telephone (518) 452-5600
Facsimile (518) 452-5579

cc:

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

In Re Application of: Schmuck et al.

Title: DETECTING WHEN TO PREFETCH INODES AND THEN PREFETCHING INODES IN PARALLEL

Attorney Docket No.: POU9-2000-0091-US1

"EXPRESS MAIL" MAILING LABEL NO. EL601699713US

Date of Deposit July 18, 2000

I hereby certify that this paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and addressed to:

BOX PATENT APPLICATION
ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

JILL K. BECKER
(Typed or printed name of person mailing paper or fee)

Jill K. Becker
(Signature of person mailing paper or fee)

Enclosures:

* Utility Patent Application Transmittal Letter (3 pages) (in duplicate)
* U.S. Patent Application which includes: Specification (16 pages), 69 Claims (21 pages), Abstract (1 page)
* Five (5) sheets of Formal Drawings
* Declaration with Power of Attorney (unsigned) (3 pages)
* Information Disclosure Citation (1 page) and seven (7) references
* Two (2) Acknowledgment Postcards

DETECTING WHEN TO PREFETCH INODES AND THEN PREFETCHING INODES IN PARALLEL

Technical Field

This invention relates, in general, to parallel data processing, and in particular, to the parallel prefetching of file meta data.

Background Art

A parallel, shared disk file environment includes a set of computer nodes, disk storage devices, a communications network, and a parallel file system running on each computer node. A parallel file system differs from a traditional distributed file system, like the Network File System (NFS) or the Distributed File System (DFS), in that with a parallel file system, data belonging to the same file is distributed or "striped" across disks that are attached to different nodes in the environment or directly attached to the network. A parallel file system allows data to be transferred between disks and computer nodes without requiring all the data to pass through a single server node.

The meta data of files, which includes the file attributes, such as file size, last-modified time, and file owner, are also striped across the disks in a parallel file system. That is, the various data structures that include the meta data (referred to as inodes) are stored on different disks.

Applications executing in a computing environment, regardless of whether the environment employs a traditional or parallel file system, often request a directory listing of the files of a directory including the file attributes. In order to provide this listing, the file system reads all of the inodes of the files of the requested directory. However, for a large directory, reading inodes one at a time can be very time consuming.

In traditional file systems, the problem of reading inodes efficiently has been addressed by clustering inodes. That is, by arranging for inodes of files of the same directory to be close together on disk (e.g., grouped together in inode blocks). Thus, instead of reading individual inodes, a whole block of inodes is read in a single I/O. Since inodes are typically small, the cost of reading a block of inodes is not much higher than reading a single inode, and reading a whole block of inodes is significantly faster than reading each inode individually.

However, this solution is not well-suited for a parallel file system for at least the following reasons:

1. Applications running on a parallel file system may concurrently access different inodes of the same directory (for example, a parallel mail server). If all of these inodes are clustered within the same inode block, then all I/Os to read or write these inodes will go to the same disk, causing access to these inodes to become a bottleneck.

2. A parallel file system requires distributed locking to synchronize access to file data and meta data from multiple nodes in the network. To read a whole block of inodes would require getting a lock on each of the inodes in the block, requiring messages to a lock coordinator. Therefore, in a parallel file system, the cost of reading a whole block of inodes is significantly higher than the cost of reading a single inode. Hence, an approach that always caches only whole inode blocks would speed up inode access only if the locking granularity were increased, so that each lock pertains to a whole block of inodes instead of an individual inode. However, this would significantly increase the number of lock conflicts due to "false sharing" between nodes: If two nodes were concurrently updating different inodes within the same inode block, then each inode update would require messages and possibly I/O to revoke the lock on the inode block from the other node.

Thus, a need still exists for an efficient technique for reading inodes of a parallel file system. In particular, a need exists for a facility that manages when and how to prefetch inodes.

Summary of the Invention

The shortcomings of the prior art are overcome and additional advantages are provided through the provision of

a method of managing the prefetching of data of files. The method includes, for instance, detecting a pattern of requests for data of multiple files; and prefetching data of a plurality of files, in response to the detecting indicating the pattern.

In another aspect of the present invention, a method of managing the prefetching of data is provided. The method includes, for instance, controlling the prefetching of data of a plurality of files by pacing at least the initiating of the prefetching based upon requests for data; and prefetching the data of the plurality of files, in response to the controlling.

In yet a further aspect of the present invention, a method of managing the prefetching of inodes associated with files of a directory is provided. The directory includes one or more directory blocks and each directory block has associated therewith zero or more files. The method includes, for example, detecting a pattern of requests for multiple inodes associated with multiple files of a directory block of the one or more directory blocks; and prefetching a plurality of inodes associated with the directory block, in response to detecting the pattern.

System and computer program products corresponding to the above-summarized methods are also described and claimed herein.

The prefetching capabilities of the present invention advantageously increase the speed at which inodes are read.

In one aspect of the present invention, a technique is provided for deciding when to prefetch data (e.g., inodes) by detecting access patterns that would benefit from such a prefetch. In a further aspect of the present invention, inodes are prefetched (at least some in parallel) at an average rate that substantially matches the speed at which an application requests file attribute data.

Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

Brief Description of the Drawings

The subject matter which is regarded as the invention is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the invention are apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

FIG. 1 depicts one example of a computing environment incorporating and using one or more aspects of the present invention;

FIG. 2 depicts further details of a node of FIG. 1, in accordance with an aspect of the present invention;

FIG. 3 depicts one example of a directory partitioned into a plurality of directory blocks, each directory block having zero or more directory entries, and each entry having associated therewith an inode, in accordance with an aspect of the present invention;

FIG. 4 depicts one embodiment of the logic used to collect inode access statistics and to start inode prefetch, in accordance with an aspect of the present invention;

FIG. 5 depicts one example of various statistics associated with each directory block of FIG. 3, in accordance with an aspect of the present invention; and

FIG. 6 depicts one embodiment of the logic used to prefetch inodes in parallel, in accordance with an aspect of the present invention.

Best Mode for Carrying Out the Invention

In accordance with an aspect of the present invention, a capability is provided for deciding when to prefetch data (such as, meta data) by detecting access patterns that would benefit from such a prefetch. Further, in another aspect of the present invention, a capability is provided for prefetching data (e.g., meta data) at an average rate that substantially matches the speed at which an application

requests data. In one example, at least some of the data is prefetched in parallel.

As used herein, data refers to any information associated with or located within a file. Further, file is used to refer to any type of entity that holds information or has information associated therewith.

One embodiment of a computing environment incorporating and/or using aspects of the present invention is described with reference to FIG. 1. Computing environment 100 includes one or more nodes 102 (e.g., Node 1,...Node n), which share access to one or more storage devices 104 (e.g., Disk 1...Disk m). The nodes are coupled to each other and to the storage devices via an interconnect 106. In one example, the interconnect includes a wire connection, token ring or network connection, to name just a few examples. One communications protocol used by one or more of these connections is TCP/IP.

As one example, a node 102 includes an operating system 200 (FIG. 2), such as the AIX operating system, offered by International Business Machines Corporation. The operating system includes a file system 202 (e.g., a software layer), such as the General Parallel File System (GPFS), offered by International Business Machines Corporation, which is used to manage one or more files located in the various storage devices.

In one example, each file is associated with a directory and in particular, with a directory block of a

directory. Thus, as shown in FIG. 3, a directory 300 includes one or more directory blocks 302, and each directory block 302 has associated therewith zero or more files 304. (A computing environment may include one or more directories.)

Further, each file has associated therewith meta data 306 that includes, for example, file attributes, such as file size, last-modified time, and file owner. This meta data is contained within a data structure, referred to as an inode.

One aspect of managing files includes the management of the data associated with the files, including the management of the meta data or inodes. This management includes, for instance, increasing the speed at which applications read inodes by detecting when prefetching of inodes is to occur, and then prefetching inodes by issuing I/O requests to multiple storage devices (e.g., disks) in parallel.

In one aspect of the present invention, the decision to prefetch inodes is based on detecting access patterns that would benefit from such a prefetch. Most file systems do not provide an interface that allows retrieving attributes from multiple files in a single call. For example, the UNIX Standard file system interface provides a readdir call, which returns the names of the files stored in the directory, and a stat call which takes the name of a single file and returns the attributes of that file, but does not provide a single call that returns the attributes of all files of a directory. This means that applications written

for a UNIX or UNIX-like file system cannot communicate to the file system their intent to retrieve attributes for all files of a directory. Instead, in order to support such applications efficiently, in accordance with an aspect of the present invention, the file system is to infer that fact from a sequence of file system calls made by the application.

One embodiment of the logic employed to determine when to prefetch data (such as meta data of inodes) is described with reference to FIG. 4. In one example, this logic is employed by one or more file systems.

Referring to FIG. 4, initially, a stat call is issued by an application, STEP 400. When the stat request is issued, the file name indicated in the request is looked up in the current directory indicated by the application, STEP 402. If the file name is found, the directory entry contains a reference to the inode (e.g., an inode number) that stores the attributes of the file.

From this information, an in-memory control structure for accessing the file is built, STEP 404. This control structure is referred to as a vnode. Recorded within the vnode is, for example, a reference to the directory and the logical block number of the directory block in which the file name was found.

Thereafter, a determination is made as to whether the inode is cached in memory, INQUIRY 406. If the inode has been accessed in the past, it still might be cached in

memory. If it is cached in memory, then cache hit statistics for the directory block are updated, STEP 408. In particular, there are various statistics associated with each directory block. For example, as shown in FIG. 5, each directory block 500 has associated therewith various statistics 502, including cache hit statistics 504 and cache miss statistics 506. Cache miss statistics 506 include a counter 508 and a timestamp 510, which are described below.

Returning to FIG. 4, subsequent to updating the cache hit statistics for the directory block, the file attributes from the inode are returned, STEP 410.

Returning to INQUIRY 406, if the inode is not cached in memory, then cache miss statistics 506 (FIG. 5) for the directory block are updated, STEP 412. In particular, if the inode of the file was not found in the cache, the corresponding statistics counter 508 associated with that directory block is incremented. If timestamp 510 associated with that counter is older than a predetermined time interval I (e.g., one second or so), the counter is reset to zero before it is incremented. Then, the current time is recorded in the timestamp. Thus, a counter value of n associated with a directory block b means that n times within a time period of length at most nxI, an inode for a file in b needed to be read from disk.

Thereafter, a determination is made as to whether a cache miss threshold has been exceeded, INQUIRY 414. For example, the value of counter 508 is compared to the cache miss threshold (e.g., 5). If the cache miss threshold has

not been exceeded, then the inode is read from disk into memory, STEP 416 (FIG. 4), and the file attributes are returned, STEP 410.

Otherwise, when counter 508 (FIG. 5) exceeds the predetermined threshold value, INQUIRY 414 (FIG. 4), this is an indication that the application is likely to request attributes for all (or at least a large subset) of the files of the directory, and that most or all of the inodes of these files are not currently cached. Hence, this is an indication that this application would benefit from inode prefetch.

Thus, inode prefetch is initiated for the given directory block, STEP 418, and the requested inode is read from disk into memory, STEP 416. Thereafter, the file attributes are returned, STEP 410. (In another example, the requested inode is also read as part of the prefetching, described below.)

One embodiment of the logic associated with prefetching inodes is described with reference to FIG. 6. In one embodiment, this logic is performed by one or more file systems.

Referring to FIG. 6, when one of the statistic counters reaches its threshold value, the prefetching of inodes is started, STEP 600. For example, the directory block that the counter is associated with is accessed and a list of inode numbers for all (or a subset) of the files of that directory block is extracted, STEP 602. For each inode in

the list, a check is made as to whether the inode is currently cached. If the inode is cached, then the prefetch continues on to the next inode. However, if the inode is not cached, then the inode is read from disk, as described herein.

In accordance with an aspect of the present invention, a plurality of inodes are read from a plurality of disks (or storage devices) in parallel. In particular, up to some number p of I/Os are started in parallel to read the first p not cached inodes of the directory block, STEP 604 (FIG. 6). (P is the degree of parallelism and is a predetermined value based on the number of disks in the file system (e.g., two times the number of disks).) Each time one of the p I/O requests completes, STEP 606, a determination is made as to whether there are more inodes to be read for the current directory block that are not yet cached, INQUIRY 608. If there are more inodes to be read, then one or more other inode I/O requests are issued, STEP 610.

Since inodes are striped across the disks, reading inodes in parallel will be up to k times faster than reading them one at a time, where k is the number of disks. As the application continues to request file attributes out of the same directory block, the file system will be able to satisfy most of these requests out of the cache. The net effect will be that attributes will be returned to the application up to k times faster than without inode prefetch.

Returning to INQUIRY 608, if there are no more inodes to be read for the current directory block, then cache hit statistics 504 (FIG. 5) for the current directory block are checked, STEP 612 (FIG. 6). These statistics are used to determine when to prefetch inodes out of the next directory block. That is, once all of the inodes associated with one directory block have been prefetched, a decision is made as whether to prefetch inodes associated with another directory block of the directory. In order to make this decision, a check is made as to the current cache hit statistics for the current directory, STEP 612. If this number is below a cache hit threshold (e.g., a predetermined fraction of the total number of files of the directory block), INQUIRY 614, then the inode prefetch will wait for the application to catch up, STEP 616.

However, once the stat count has reached the predetermined fraction, INQUIRY 614, a determination is made as to whether there are more blocks in the directory, INQUIRY 618. If not, then prefetching is complete, STEP 620. However, if there are more blocks, then inode prefetch continues by reading inodes for files of the next directory block, STEP 622. This is done in the manner described herein.

Using the above-described technique, inodes are prefetched in parallel, one directory block at a time, at an average rate that substantially matches the speed at which the application accesses these inodes. For example, if a fraction of 50% is chosen, this technique ensures that not more than 1½ directory blocks worth of inodes are prefetched

ahead of the application. In particular, once the prefetching of inodes from one directory block is completed, it is desirable to start prefetching inodes from the next directory block before the application starts requesting attributes of those inodes. Otherwise, if the access pattern detection mechanism described above was relied upon to trigger the inode prefetch for the next directory block, then for every directory block there would be a delay to reach the necessary threshold before the remaining inodes would be read. Depending on parameters, such as average file name length, directory block size, etc., the number of files per directory block may be small enough for this delay to significantly reduce the speed-up that could otherwise be had from the parallel inode prefetch. On the other hand, simply prefetching all of the remaining inodes in the directory as fast as possible would also not be satisfactory, because the total number of files of the directory may be larger than the number of inodes that fit in the cache. Prefetching inodes faster than the application issues stat calls would cause inode prefetch to throw inodes out of the cache that had been prefetched earlier, but have not yet been accessed by the application. This would negate any benefits of the inode prefetch.

Described in detail above is one embodiment of efficiently determining when to prefetch inodes and the prefetching of those inodes in parallel. Although the embodiments described herein refer to inodes, the capabilities of the present invention are not limited to inodes. One or more aspects of the present invention can be employed in reading other types of data.

The capabilities of the present invention are well suited for many types of situations and/or applications, including, for instance, applications that recursively traverse a directory tree, reading part of a directory, descending into a subdirectory, and then resume reading the parent directory upon returning from the subdirectory. The pacing mechanism described herein would suspend the inode prefetch for the parent directory, while its subdirectory is being processed, prefetch inodes for the subdirectory as necessary, and then resume prefetching inodes from the parent once the application continues accessing the parent directory.

The above-described computing environment is offered as only one example. One or more aspects of the present invention can be incorporated and used with many types of computing units, computers, processors, nodes, systems, workstations and/or environments without departing from the spirit of the present invention.

The present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The article of manufacture can be included as a part of a computer system or sold separately.

Additionally, at least one program storage device readable by a machine, tangibly embodying at least one

program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

The flow diagrams depicted herein are just examples. There may be many variations to these diagrams or the steps (or operations) described therein without departing from the spirit of the invention. For instance, the steps may be performed in a differing order, or steps may be added, deleted or modified. All of these variations are considered a part of the claimed invention.

Although preferred embodiments have been depicted and described in detail herein, it will be apparent to those skilled in the relevant art that various modifications, additions, substitutions and the like can be made without departing from the spirit of the invention and these are therefore considered to be within the scope of the invention as defined in the following claims.

Claims

What is claimed is:

1. A method of managing the prefetching of data of files, said method comprising:

detecting a pattern of requests for data of multiple files; and

prefetching data of a plurality of files, in response to said detecting indicating said pattern.

2. The method of claim 1, wherein said data comprises meta data.

3. The method of claim 1, wherein said multiple files and said plurality of files are associated with a single directory.

4. The method of claim 1, wherein said detecting indicates said pattern when said detecting determines that a predefined number of requests for data could not be satisfied by reading a cache.

5. The method of claim 1, wherein said detecting comprises determining whether a cache miss threshold has been exceeded, wherein said detecting indicates said pattern when said cache miss threshold has been exceeded.

6. The method of claim 5, wherein said determining comprises comparing a counter of cache misses that occurred within a preselected time interval to said cache miss threshold to determine whether said cache miss threshold has been exceeded.

7. The method of claim 6, wherein said counter and said cache miss threshold are associated with a directory block of a directory of files, said directory of files comprising said multiple files and said plurality of files, and said directory of files comprising one or more directory blocks.

8. The method of claim 1, wherein said prefetching comprises prefetching data of at least some files of said plurality of files in parallel.

9. The method of claim 1, wherein said prefetching comprises prefetching data at an average rate that substantially matches a speed of requests for data.

10. The method of claim 1, wherein said prefetching comprises:

obtaining data associated with a number of files of said plurality of files;

determining whether a cache hit threshold has been reached; and

obtaining data associated with one or more additional files of said plurality of files, in response to reaching said cache hit threshold.

11. The method of claim 10, wherein said obtaining data associated with said number of files comprises issuing a plurality of I/O requests to read data of at least a portion of said number of files in parallel.

12. A method of managing the prefetching of data, said method comprising:

controlling the prefetching of data of a plurality of files by pacing at least the initiating of the prefetching based upon requests for data; and

prefetching said data of said plurality of files, in response to said controlling.

13. The method of claim 12, wherein said controlling comprises determining whether a cache hit threshold has been reached, wherein said prefetching is performed in response to reaching said cache hit threshold.

14. The method of claim 12, wherein said prefetching comprises prefetching data of at least some files of said plurality of files in parallel.

15. The method of claim 12, further comprising detecting a pattern of requests for data of multiple files, wherein said pattern indicates prefetching is to occur.

16. The method of claim 12, wherein said data comprises meta data.

17. A method of managing the prefetching of inodes associated with files of a directory, said directory comprising one or more directory blocks and each directory block having associated therewith zero or more files, said method comprising:

detecting a pattern of requests for multiple inodes associated with multiple files of a directory block of said one or more directory blocks; and

prefetching a plurality of inodes associated with said directory block, in response to detecting said pattern.

18. The method of claim 17, wherein said directory block has associated therewith a counter and a cache miss threshold, said counter representing a number of inodes associated with said directory block that were requested within a preselected amount of time and were not found in a cache, and wherein said detecting comprises comparing said counter to said cache miss threshold to determine whether said pattern exists.

19. The method of claim 17, wherein said prefetching comprises prefetching at least a portion of said plurality of inodes in parallel.

20. The method of claim 17, further comprising initiating the prefetching of one or more inodes associated with another directory block of said directory, wherein said initiating is in response to requests for inodes of said directory.

21. The method of claim 20, wherein said initiating comprises determining whether a cache hit threshold has been reached, wherein said prefetching of one or more inodes associated with said another directory block is initiated when said cache hit threshold is reached.

22. A system of managing the prefetching of data of files, said system comprising:

means for detecting a pattern of requests for data of multiple files; and

means for prefetching data of a plurality of files, in response to said means for detecting indicating said pattern.

23. The system of claim 22, wherein said data comprises meta data.

24. The system of claim 22, wherein said multiple files and said plurality of files are associated with a single directory.

25. The system of claim 22, wherein said means for detecting indicates said pattern when said means for detecting determines that a predefined number of requests for data could not be satisfied by reading a cache.

26. The system of claim 22, wherein said means for detecting comprises means for determining whether a cache miss threshold has been exceeded, wherein said means for detecting indicates said pattern when said cache miss threshold has been exceeded.

27. The system of claim 26, wherein said means for determining comprises means for comparing a counter of cache misses that occurred within a preselected time interval to said cache miss threshold to determine whether said cache miss threshold has been exceeded.

28. The system of claim 27, wherein said counter and said cache miss threshold are associated with a directory block of a directory of files, said directory of files comprising said multiple files and said plurality of files, and said directory of files comprising one or more directory blocks.

29. The system of claim 22, wherein said means for prefetching comprises means for prefetching data of at least some files of said plurality of files in parallel.

30. The system of claim 22, wherein said means for prefetching comprises means for prefetching data at an average rate that substantially matches a speed of requests for data.

31. The system of claim 22, wherein said means for prefetching comprises:

means for obtaining data associated with a number of files of said plurality of files;

means for determining whether a cache hit threshold has been reached; and

means for obtaining data associated with one or more additional files of said plurality of files, in response to reaching said cache hit threshold.

32. The system of claim 31, wherein said means for obtaining data associated with said number of files comprises means for issuing a plurality of I/O requests to read data of at least a portion of said number of files in parallel.

33. A system of managing the prefetching of data, said system comprising:

means for controlling the prefetching of data of a plurality of files by pacing at least the initiating of the prefetching based upon requests for data; and

means for prefetching said data of said plurality of files, in response to the controlling.

34. The system of claim 33, wherein said means for controlling comprises means for determining whether a cache hit threshold has been reached, wherein prefetching is performed in response to reaching said cache hit threshold.

35. The system of claim 33, wherein said means for prefetching comprises means for prefetching data of at least some files of said plurality of files in parallel.

36. The system of claim 33, further comprising means for detecting a pattern of requests for data of multiple files, wherein said pattern indicates prefetching is to occur.

37. The system of claim 33, wherein said data comprises meta data.

38. A system of managing the prefetching of inodes associated with files of a directory, said directory comprising one or more directory blocks and each directory block having associated therewith zero or more files, said system comprising:

means for detecting a pattern of requests for multiple inodes associated with multiple files of a directory block of said one or more directory blocks; and

means for prefetching a plurality of inodes associated with said directory block, in response to detecting said pattern.

39. The system of claim 38, wherein said directory block has associated therewith a counter and a cache miss threshold, said counter representing a number of inodes associated with said directory block that were requested within a preselected amount of time and were not found in a cache, and wherein said means for detecting comprises means for comparing said counter to said cache miss threshold to determine whether said pattern exists.

40. The system of claim 38, wherein said means for prefetching comprises means for prefetching at least a portion of said plurality of inodes in parallel.

41. The system of claim 38, further comprising means for initiating the prefetching of one or more inodes associated with another directory block of said directory, wherein the initiating is in response to requests for inodes of said directory.

42. The system of claim 41, wherein said means for initiating comprises means for determining whether a cache hit threshold has been reached, wherein the prefetching of one or more inodes associated with said another directory block is initiated when said cache hit threshold is reached.

43. A system of managing the prefetching of data of files, said system comprising:

a first node adapted to detect a pattern of requests for data of multiple files; and

at least one second node adapted to prefetch data of a plurality of files, in response to the detecting indicating said pattern.

44. The system of claim 43, wherein said at least one second node includes said first node.

45. A system of managing the prefetching of data, said system comprising:

a first node adapted to control the prefetching of data of a plurality of files by pacing at least the initiating of the prefetching based upon requests for data; and

at least one second node adapted to prefetch said data of said plurality of files, in response to the controlling.

46. The system of claim 45, wherein said at least one second node includes said first node.

47. A system of managing the prefetching of inodes associated with files of a directory, said directory comprising one or more directory blocks and each directory block having associated therewith zero or more files, said system comprising:

a first node adapted to detect a pattern of requests for multiple inodes associated with multiple files of a directory block of said one or more directory blocks; and

at least one second node adapted to prefetch a plurality of inodes associated with said directory block, in response to detecting said pattern.

48. The system of claim 47, wherein said at least one second node includes said first node.

49. At least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform a method of managing the prefetching of data of files, said method comprising:

detecting a pattern of requests for data of multiple files; and

prefetching data of a plurality of files, in response to said detecting indicating said pattern.

50. The at least one program storage device of claim 49, wherein said data comprises meta data.

51. The at least one program storage device of claim 49, wherein said multiple files and said plurality of files are associated with a single directory.

52. The at least one program storage device of claim 49, wherein said detecting indicates said pattern when said detecting determines that a predefined number of requests for data could not be satisfied by reading a cache.

53. The at least one program storage device of claim 49, wherein said detecting comprises determining whether a cache miss threshold has been exceeded, wherein said

detecting indicates said pattern when said cache miss threshold has been exceeded.

54. The at least one program storage device of claim 53, wherein said determining comprises comparing a counter of cache misses that occurred within a preselected time interval to said cache miss threshold to determine whether said cache miss threshold has been exceeded.

55. The at least one program storage device of claim 54, wherein said counter and said cache miss threshold are associated with a directory block of a directory of files, said directory of files comprising said multiple files and said plurality of files, and said directory of files comprising one or more directory blocks.

56. The at least one program storage device of claim 49, wherein said prefetching comprises prefetching data of at least some files of said plurality of files in parallel.

57. The at least one program storage device of claim 49, wherein said prefetching comprises prefetching data at an average rate that substantially matches a speed of requests for data.

58. The at least one program storage device of claim 49, wherein said prefetching comprises:

obtaining data associated with a number of files of said plurality of files;

determining whether a cache hit threshold has been reached; and

obtaining data associated with one or more additional files of said plurality of files, in response to reaching said cache hit threshold.

59. The at least one program storage device of claim 58, wherein said obtaining data associated with said number of files comprises issuing a plurality of I/O requests to read data of at least a portion of said number of files in parallel.

60. At least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform a method of managing the prefetching of data, said method comprising:

controlling the prefetching of data of a plurality of files by pacing at least the initiating of the prefetching based upon requests for data; and

prefetching said data of said plurality of files, in response to said controlling.

61. The at least one program storage device of claim 60, wherein said controlling comprises determining whether a cache hit threshold has been reached, wherein said prefetching is performed in response to reaching said cache hit threshold.

62. The at least one program storage device of claim 60, wherein said prefetching comprises prefetching data of at least some files of said plurality of files in parallel.

63. The at least one program storage device of claim 60, wherein said method further comprises detecting a pattern of requests for data of multiple files, wherein said pattern indicates prefetching is to occur.

64. The at least one program storage device of claim 60, wherein said data comprises meta data.

65. At least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform a method of managing the prefetching of inodes associated with files of a directory, said directory having associated therewith one or more directory blocks and each directory block comprising zero or more files, said method comprising:

detecting a pattern of requests for multiple inodes associated with multiple files of a directory block of said one or more directory blocks; and

prefetching a plurality of inodes associated with said directory block, in response to detecting said pattern.

66. The at least one program storage device of claim 65, wherein said directory block has associated therewith a counter and a cache miss threshold, said counter representing a number of inodes associated with said directory block that were requested within a preselected amount of time and were not found in a cache, and wherein said detecting comprises comparing said counter to said cache miss threshold to determine whether said pattern exists.

67. The at least one program storage device of claim 65, wherein said prefetching comprises prefetching at least a portion of said plurality of inodes in parallel.

68. The at least one program storage device of claim 65, wherein said method further comprises initiating the prefetching of one or more inodes associated with another directory block of said directory, wherein said initiating is in response to requests for inodes of said directory.

69. The at least one program storage device of claim 68, wherein said initiating comprises determining whether a cache hit threshold has been reached, wherein said prefetching of one or more inodes associated with said another directory block is initiated when said cache hit threshold is reached.

* * * * *

DETECTING WHEN TO PREFETCH INODES AND THEN PREFETCHING INODES IN PARALLEL

Abstract of the Disclosure

The decision to prefetch inodes is based upon the detecting of access patterns that would benefit from such a prefetch. Once the decision to prefetch is made, a plurality of inodes are prefetched in parallel. Further, the prefetching of inodes is paced, such that the prefetching substantially matches the speed at which an application requests inodes.

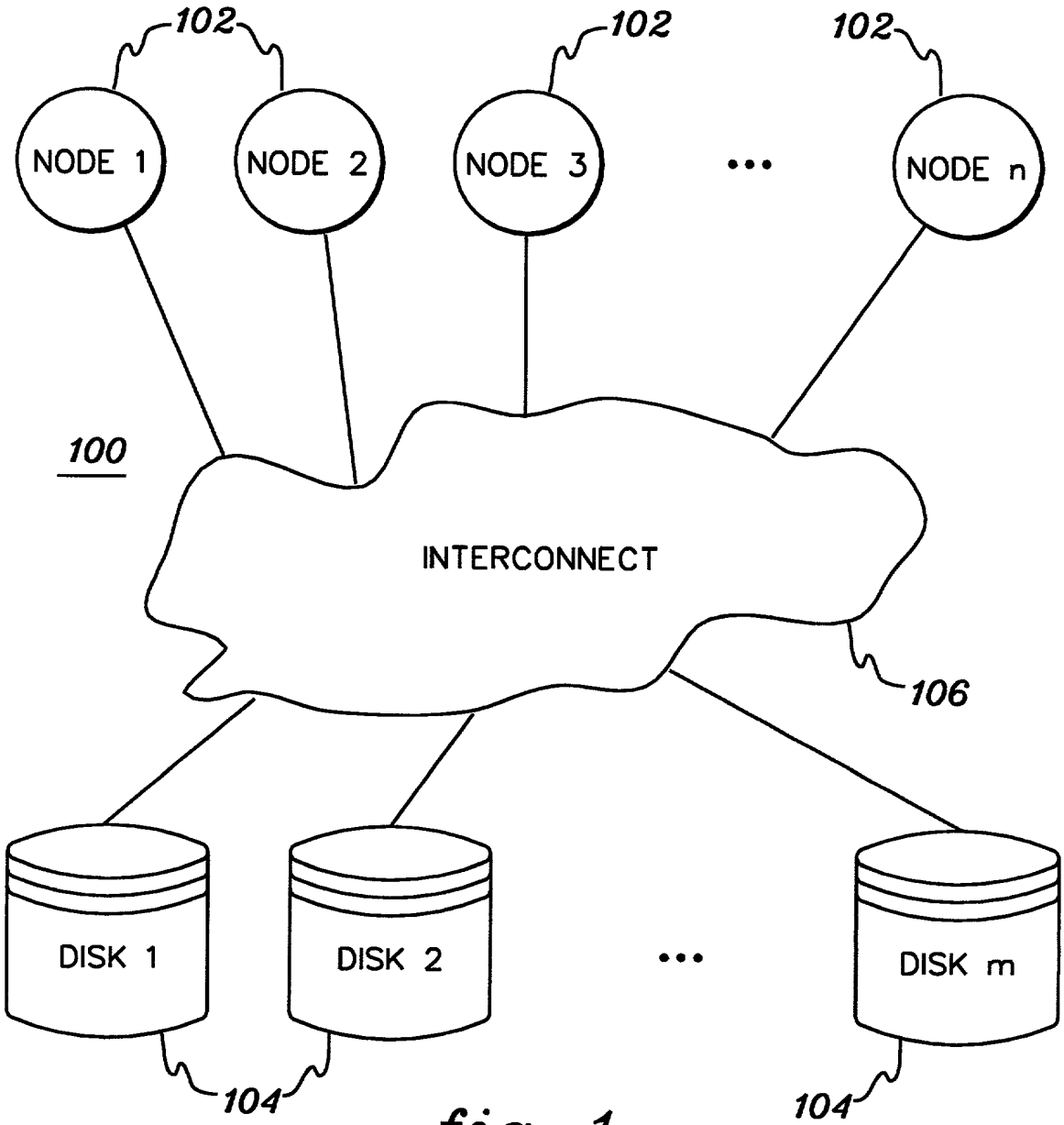
102
102
102
NODE 1
NODE 2
NODE 3
...
NODE n
100
INTERCONNECT
106
DISK 1
DISK 2
...
DISK m
104
104

fig. 1

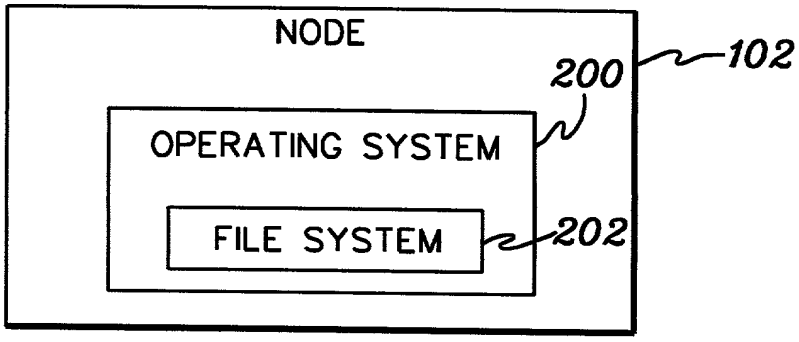
NODE
102
200
OPERATING SYSTEM
FILE SYSTEM
202

fig. 2

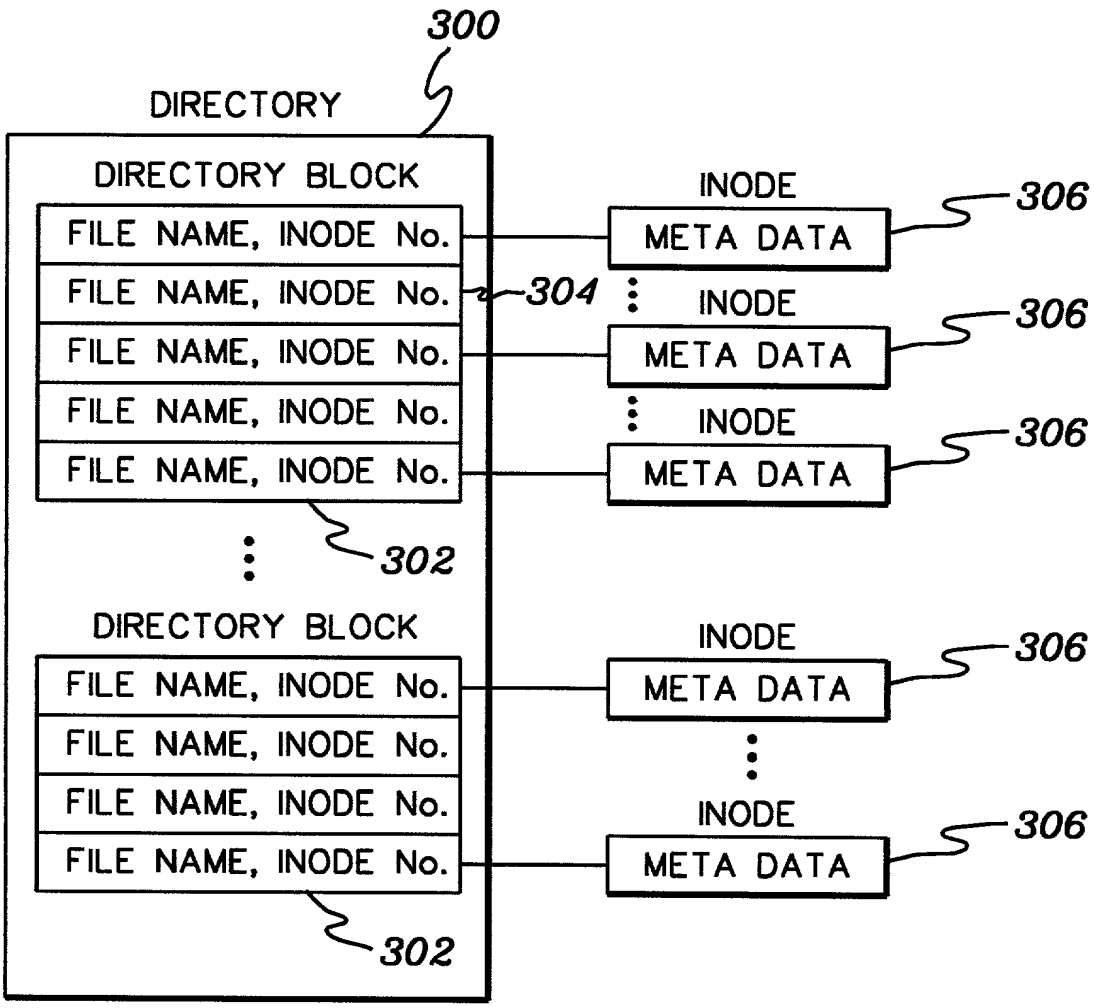
300
DIRECTORY
DIRECTORY BLOCK
FILE NAME, INODE No.
FILE NAME, INODE No.
FILE NAME, INODE No.
FILE NAME, INODE No.
FILE NAME, INODE No.
304
302
DIRECTORY BLOCK
FILE NAME, INODE No.
FILE NAME, INODE No.
FILE NAME, INODE No.
FILE NAME, INODE No.
302
INODE
META DATA
306
INODE
META DATA
306
INODE
META DATA
306
INODE
META DATA
306
INODE
META DATA
306

fig. 3

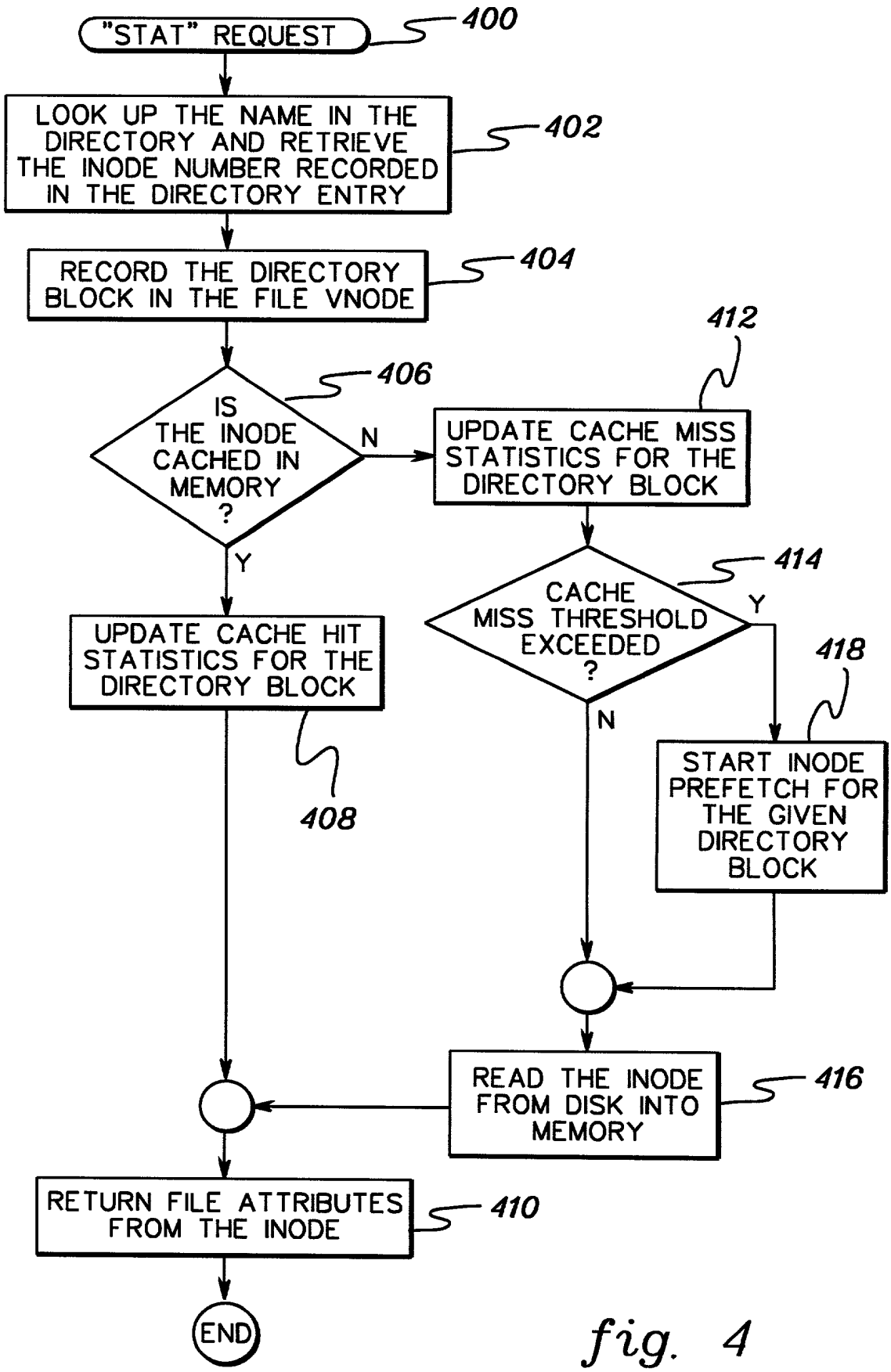
"STAT" REQUEST
400
LOOK UP THE NAME IN THE DIRECTORY AND RETRIEVE THE INODE NUMBER RECORDED IN THE DIRECTORY ENTRY
402
RECORD THE DIRECTORY BLOCK IN THE FILE VNODE
404
IS THE INODE CACHED IN MEMORY ?
406
N
Y
UPDATE CACHE HIT STATISTICS FOR THE DIRECTORY BLOCK
408
UPDATE CACHE MISS STATISTICS FOR THE DIRECTORY BLOCK
412
CACHE MISS THRESHOLD EXCEEDED ?
414
Y
N
START INODE PREFETCH FOR THE GIVEN DIRECTORY BLOCK
418
READ THE INODE FROM DISK INTO MEMORY
416
RETURN FILE ATTRIBUTES FROM THE INODE
410
END

fig. 4

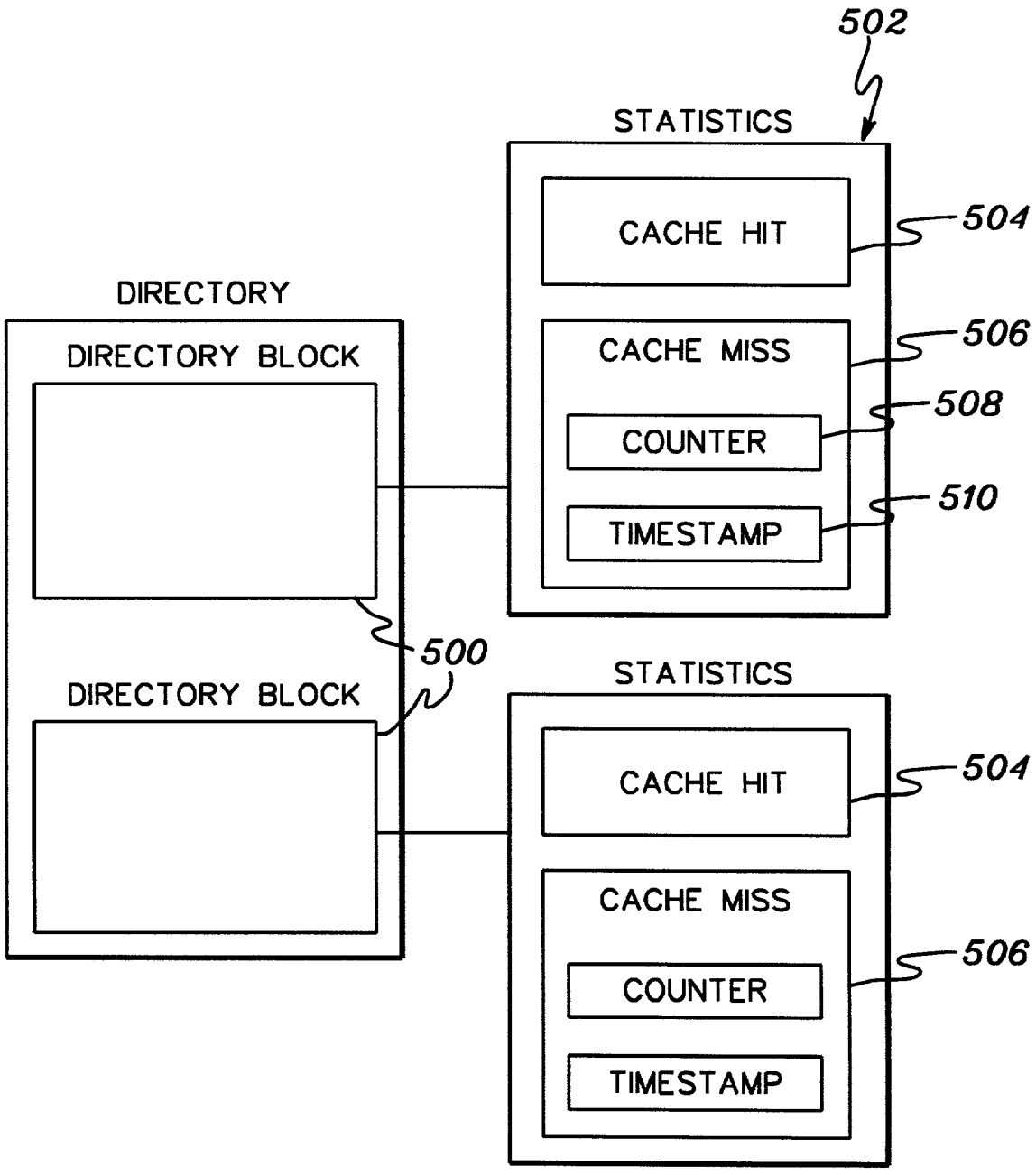
502
STATISTICS
CACHE HIT
504
DIRECTORY
DIRECTORY BLOCK
CACHE MISS
506
508
COUNTER
510
TIMESTAMP
500
DIRECTORY BLOCK
STATISTICS
CACHE HIT
504
CACHE MISS
506
COUNTER
TIMESTAMP

fig. 5

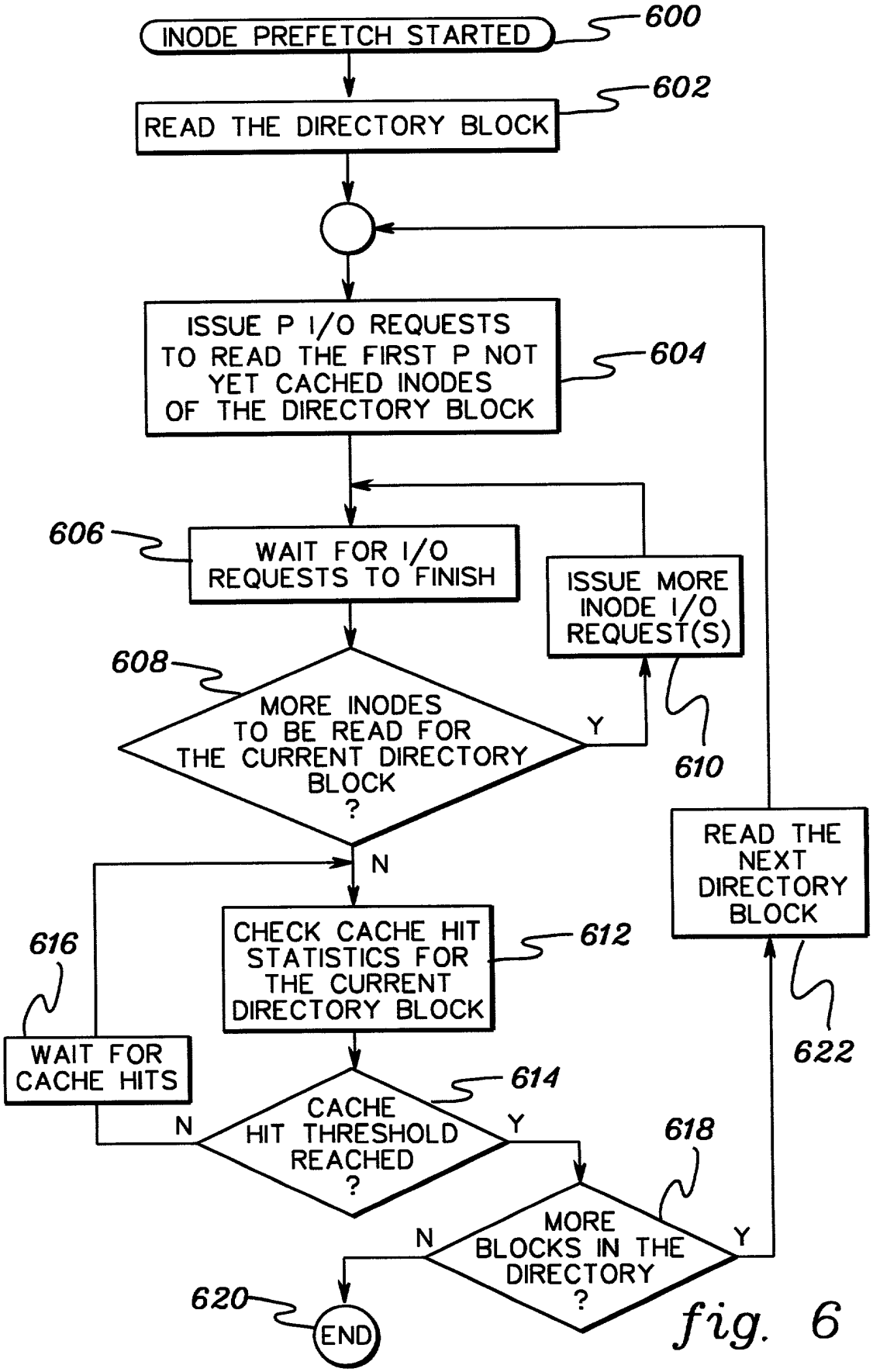
INODE PREFETCH STARTED
600
READ THE DIRECTORY BLOCK
602
ISSUE P I/O REQUESTS TO READ THE FIRST P NOT YET CACHED INODES OF THE DIRECTORY BLOCK
604
606
WAIT FOR I/O REQUESTS TO FINISH
ISSUE MORE INODE I/O REQUEST(S)
610
608
MORE INODES TO BE READ FOR THE CURRENT DIRECTORY BLOCK ?
Y
N
READ THE NEXT DIRECTORY BLOCK
622
CHECK CACHE HIT STATISTICS FOR THE CURRENT DIRECTORY BLOCK
612
616
WAIT FOR CACHE HITS
614
CACHE HIT THRESHOLD REACHED ?
N
Y
618
MORE BLOCKS IN THE DIRECTORY ?
N
Y
620
END

fig. 6

Docket No.
POU9-2000-0091-US1

Declaration and Power of Attorney For Patent Application

English Language Declaration

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

DETECTING WHEN TO PREFETCH INODES AND THEN PREFETCHING INODES IN PARALLEL

the specification of which

(check one)

☒ is attached hereto.

☐ was filed on _____ as United States Application No. or PCT International Application Number _____

and was amended on _____

(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d) or Section 365(b) of any foreign application(s) for patent or inventor's certificate, or Section 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate or PCT International application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s)			Priority Not Claimed
_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	☐
_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	☐
_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	☐

I hereby claim the benefit under 35 U.S.C. Section 119(e) of any United States provisional application(s) listed below:

(Application Serial No.)	(Filing Date)

I hereby claim the benefit under 35 U. S. C. Section 120 of any United States application(s), or Section 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 U.S.C. Section 112, I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known to me to be material to patentability as defined in Title 37, C. F. R., Section 1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application:

(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. *(list name and registration number)*

Lynn L. Augspurger, Reg. No. 24,227
Lawrence D. Cutter, Reg. No. 28,501
Marc A. Ehrlich, Reg. No. 39, 966
William B. Porter, Reg. No. 33,135
Floyd A. Gonzalez, Reg. No. 26,732
William A. Kinnaman, Jr., Reg. No. 27,650
Lily Neff, Reg. No. 38,254
Andrew J. Wojnicki, Jr., Reg. No. 43,995
Christopher A. Hughes, Reg. No. 26,914
Edward A. Pennington, Reg. No. 32,588
John E. Hoel, Reg. No. 26,279
Joseph C. Redmond, Jr., Reg. No. 18,753
Jeff Rothenberg, Reg. No. 26,429
Kevin P. Radigan, Reg. No. 31,789
Blanche E. Schiller, Reg. No. 35,670

Send Correspondence to: **Blanche E. Schiller, Esq.**
HESLIN & ROTHENBERG, P.C.
5 Columbia Circle
Albany, NY 12203

Direct Telephone Calls to: *(name and telephone number)*
Blanche E. Schiller, Esq. (518) 452-5600

Full name of sole or first inventor **FRANK B. SCHMUCK**
Sole or first inventor's signature Date
Residence **406-A Union Avenue, Campbell, CA 95008**
Citizenship **German**
Post Office Address **406-A Union Avenue, Campbell, CA 95008**

Full name of second inventor, if any **JAMES C. WYLLIE**
Second inventor's signature Date
Residence **18392 Chadbourne Lane, Monte Sereno, CA 95030**
Citizenship **United States of America**
Post Office Address **18392 Chadbourne Lane, Monte Sereno, CA 95030**